DOCUMENT RESUME

ED 479 578 IR 022 182

AUTHOR	Matsuno, Ryoji; Tsutsumi, Yutaka; Gilbert, Richard
TITLE	Developing Componentware for Education: A Programming Approach Providing Teachers More Creative Control of Course Design.
PUB DATE	2002-10-00
NOTE	9p.; In: E-Learn 2002 World Conference on E-Learning in Corporate, Government, Healthcare, & Higher Education. Proceedings (7th, Montreal, Quebec, Canada, October 15-19, 2002); see IR 022 093. Figure contains illegible type.
AVAILABLE FROM	Association for the Advancement of Computing in Education (AACE), P.O. Box 3728, Norfolk, VA 23514. Tel: 757-623-7588; e-mail: info@aace.org; Web site: http://www.aace.org/DL/.
PUB TYPE	Reports - Evaluative (142) -- Speeches/Meeting Papers (150)
EDRS PRICE	EDRS Price MF01 Plus Postage. PC Not Available from EDRS
DESCRIPTORS	Computer Assisted Instruction; *Computer Software Development; *Courseware; Educational Technology; English (Second Language); *Instructional Design; Instructional Development; *Instructional Materials; *Material Development
IDENTIFIERS	Component Theory

ABSTRACT

The aim of this project is to build software development environments for teachers, enabling them to produce their own educational applications for their classrooms as easily as they might produce educational materials using a word processor. As one means of realizing this goal, the authors proposed a componentware methodology, and developed 15 components and three application programs for English as a Foreign Language (ESL), and presented the results (at ED-MEDIA 2002). They have continued researching componentware methodologies, and have created a wide variety of new components, categorizing these components into two main groups: basic components and high-level components. Basic components allow teachers to develop complicated CALL applications with a few simple programming statements. High-level components are, for example, independent applications not requiring any additional coding tasks. This paper discusses: the background of the project; advantages of componentware methodology; past results; basic components and high-level components; example applications made using these components; and related issues. (Author)

Reproductions supplied by EDRS are the best that can be made
from the original document.

ED 479 578

Developing Componentware for Education: A Programming Approach Providing Teachers More Creative Control of Course Design

Ryoji Matsuno,
Prefectural University of Kumamoto, Japan
matsuno@pu-kumamoto.ac.jp

Yutaka Tsutsumi
Kumamoto Gakuen University, Japan
yutaka@kumagaku.ac.jp

Richard Gilbert
Kumamoto University, Japan
gilbert@lct.kumamoto-u.ac.jp

Abstract: The aim of this project is to build software development environments for teachers, enabling them to produce their own educational applications for their classrooms as easily as they might produce educational materials using a word processor. As one of means of realization of this goal, we have proposed a componentware methodology, and have developed fifteen components and three application programs for English as a Foreign Language, and presented the result (at ED-MEDIA 2002). We have continued researching componentware methodologies, and have created a wide variety of new components, categorizing these components into two main groups: basic components and high-level components. Basic components allow teachers to develop complicated CALL applications with a few simple programming statements. High-level components are, for example, independent applications not requiring any additional coding tasks. In this paper, we will describe 1) the background of the project, 2) advantages of componentware methodology, 3) past results, 4) basic components and high-level components, 5) Example applications made using our components , 6)a discussion of related issues and, 7) concluding remarks.

Background

Although a large number of computer-based teaching materials currently exist in the marketplace, these materials are not prevalent in the classroom (MEXT, 2001). One important factor contributing to their lack of use may be that these materials are designed to suit the largest common-denominator of students, and thus are often unsuited to individual classroom requirements, as instructional methods (and student comprehension) are highly variable and specific to each unique educational and curricular environment.

If a teacher were able to develop teaching materials on a multimedia computer in a manner similar to the development of more conventional teaching materials, such multimedia teaching materials would be mo re easily and thus more frequently implemented. Most teachers are acquainted with their students' abilities, and would be able to develop multimedia teaching materials corresponding to their students' needs, if minimal-to-zero programming knowledge on the teacher's part were required. As a solution to the current dilemma, we are proposing a componentware methodology. In the following section, we will describe the advantage of componentware methodology in contrast to more traditional authoring tools many teachers are currently using in order to produce teaching materials.

IR022182

Advantages of componentware methodology

PERMISSION TO REPRODUCE AND DISSEMINATE THIS MATERIAL IN MICROFICHE ONLY HAS BEEN GRANTED BY

G.H. Marks

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

2B

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

☑ This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

BEST COPY AVAILABLE

In contemporary education, many teachers are using authoring tools such as HyperCard and Toolbox in order to produce teaching materials. These authoring tools provide users with software-routines encapsulating some features enabling users to handle graphics and text. By using these software-routines, teachers can produce their own teaching materials. However, such authoring tools have some severe limitations. Some of these limitations are: 1) they require some programming ability, 2) they cannot provide us with new software-routines and do not allow us to change features of their software-routines in order to fit a specific purpose, and thus, 3) it is difficult to implement in software the rapid changes in multimedia technology, and in addition 4) such programs sometimes force us to re-work teaching materials whenever upgraded or entirely new versions arrive.

Componentware methodology has a great potential to overcome the above-mentioned limitations. Componentware is an object-oriented software development method which uses and integrates components as essential building blocks (Aoyama, 1996; Jacobson et al., 1995; Udel, 1994). Componentware methodology allows us to develop software systems in both a top-down and a bottom-up fashion. This is an essential aspect of educational software systems development. The bottom-up approach can allow a teacher to easily develop a new educational software system by assembling pre-existing components. In the case where a teacher wants to develop a new type of teaching material, the top-down fashion can be adopted to break down a teacher's requirements into functionally modular components. Another advantage of the componentware methodology is in meeting requirements for computer usage in the real educational world: We can categorize teachers into three groups: an advanced group able to develop original program components, an intermediate group able to assemble pre-existing components into educational software, and a beginner group who can use pre-existing educational software systems in their classes. The advanced group develops components, making full use of their specialized knowledge and provides them to the intermediate group; the intermediate group is able to make educational software by using given components (i.e. they can produce specialized software systems without specialized knowledge); and the intermediate group is able to provide educational software to the beginner group—the beginner group is then able to make use of the educational software in their classes. If this flow from the advanced group to the beginner group is made to function smoothly, computer-based education can become more effectively and easily implemented.

Past Results

We have developed four component-based software applications and fifteen reusable software components for English as a Foreign Language (EFL) education (McCarthy, Matsuno and Swan, 1996; Matsuno, Tsutsumi, and Ushijima, 1999; Gilbert and Matsuno, 2000a and 2000b). In the EFL field, teachers have not only been using multimedia devices such as tape-recorders, CDs and videotapes, but also have been using computers since the advent of the institutional computer lab. In this field, therefore, multimedia computer systems and software offer a significant contribution to education

In a pilot study, three beginner-programmer students produced EFL applications using componentware components. They were all able to produce EFL applications with sound and movie clips as easily as producing EFL educational materials using, for instance, word-processor software, after they were given instruction on the usage and programming of each component. (Instruction occurred within one 90-minute class.) There were some common themes in the students' software products, exemplifying typical EFL materials —for example a "listen-and-repeat" practice-program, Japanese/English bilingual-switching dialogue program, etc. While these were useful products, additionally the students had wanted to produce evaluation tools documenting the learning-effect. But these types of components (e.g. quiz-type or questionnaire-type components) required more challenging programming structures. There are three typical program structures for combining statements in a program: sequencing, conditional(if statement), and repetition(loop). The students used only the sequencing-structure component-types, and had difficulty using conditional- and/or repetition-structures. In this situation we

were faced with two possibilities. The first was to instruct the students on conditional- and repetition-structures. A second possibility was to create some mechanism that would enable students to produce evaluation components without the necessity to code with conditional/repetition structures. As we have mentioned, the aim of this project has been to build software development environments enabling teachers to produce applications as easily as producing educational materials using a word processor. With this in mind, we examined our existing components, and classified them into two categories: basic components and high-level components. Both component-levels now include encapsulated conditional/repetition structures (which no longer require extra coding of these structures by the student). The difference between the two levels is that the high-level components are actually a conglomerative integration of a number of basic components, thus are oriented towards the beginner programmer (they are close to being stand-alone applications). In the following section, we will describe both basic components and high-level components.

Basic components and High-level components

Basic components (Fig. 1) allow teachers to set or change the required properties and parameters of each component during the design-stage as well as at any time the program is used. As well, the program does not have to be "run" in order for the properties to be confirmed (usually any change in design is seen immediately and visually). This means that since teachers are able to set and confirm many properties (e.g. "font color", "caption title", "audio-file filename") without programming, they are able to produce complicated applications using only a few simple lines of programming code. As well, teachers can change or set properties dynamically.
The intermediate group is able to produce higher-level applications by joining a variety of components together, using their more advanced programming skills. High-level components (Fig. 2) are component frames, and include several individual components. Both the properties and the more complex linking routines of high-level components are set in advance by the component builders (our componentware research team, or similar advanced programming group). Therefore, the beginner group can produce an application program with no programming requirements, and the intermediate group can produce a great variety of applications with the addition of extra components/frames and/or by adding new application behaviors by using inherited features of object programming. This is similar to cooking a frozen pizza. Even though you have no idea how to make a "scratch" pizza, you can microwave a frozen pizza in a few minutes by following some simple instructions; this enables you to eat pizza quickly and easily. If you have even modest cooking skill you can add some toppings, etc., prior to the actual cooking process for a more delicious frozen pizza. It may be noted that on occasion, even a talented (perhaps hungry) chef may wish to save time and effort by cooking a frozen pizza.

Basic components

Multimedia Components

The "Multimedia Components Group" is a set of components allowing teachers to use multimedia files easily. We will introduce MMLabel (Fig. 3), one of the multimedia-group components, as an example. MMLabel is helpful when a teacher wants to produce, for instance, an English Learning program for students whose mother-tongue is Japanese. Since MMLabel has two captions and a multimedia filename property, the teacher can set the English dialog text to caption1, a parallel Japanese translation-text dialog to caption2, and designate an English sound/movie clip filename (for example showing the teacher speaking). This process requires very little design time to implement. Now, all that the teacher needs to do is just press the "Run"-button. Students are able to see the caption1 text of the MMLabel component with the English dialog on the screen, and when they click on the MMLabel, they will listen to the English dialog or will be able to watch the movie clip. The teacher can use as many as MMLabels, as they wish, and can thus produce a conversation-practice program without coding. In case the teacher wishes to show students the corresponding Japanese dialog, the teacher only needs to write one line

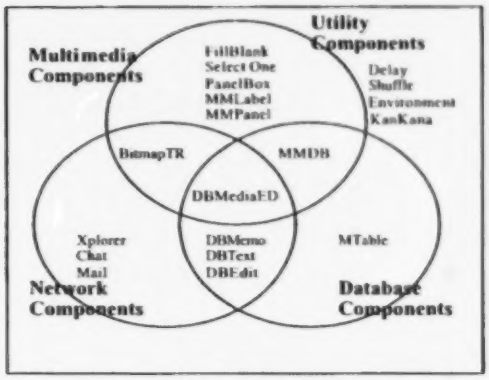

Fig 1. Basic Components

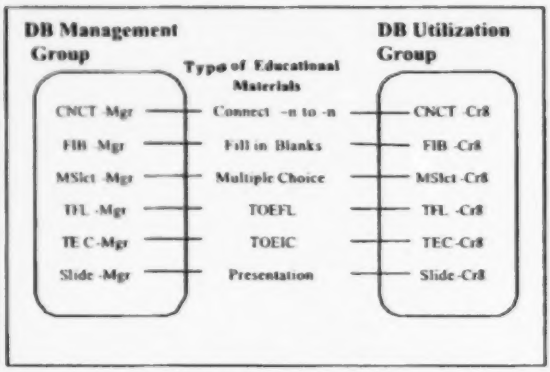

Fig 2. High-Level Components

Fig 3. An Example Application made using MMLabel

of simple code: "MMLabel.caption:= caption2". In the case of a teacher who has even modest programming ability, a variety of programs can be created in a like manner.

Communication Components

Communication components allow a teacher to make communicative programs such as their own browser and chat programs. The browser component is made using Microsoft's IE-engine. The Intermediate group can make their own browser by using the IE-Engine, but there are still difficulties for the beginner group. A similar

BEST COPY AVAILABLE

situation is faced when beginners try to make their own chat and mail programs. However, even the beginner group can make these types of programs very easily within the componentware environment. Additionally, since componentware browser components can show web pages in a specified order and at specified intervals, teachers can easily incorporate slide shows into their program materials.

Database components

The Database components group is a set of compound components handling multimedia and internet functions. We will introduce the MMTable component as an example. This component allows for the concurrent use of several different events, for instance: the display of a caption, display of an image, playing of a sound/movie. As well, this component can convert selected database elements into html form. The MMTable component has two important properties for easy database programming, a DatabaseName and a MultimediaField property. All a teacher has to do is set the database-name property to a DatabaseName parameter, and set the database field parameter of the MultimediaField property during the design phase. For example, set DatabaseName to "English Media" and set the multimedia field property to "Record I". The teacher can then use the specified database elements, and can additionally create new elements if necessary, without needing to re-design their program.

Utility Components

When making educational programs, we have observed that similar functions are often utilized (for example: shuffling/randomizing, delay-interval timer, kanji-to-kana converter). Utility components are sets of such small "gadget programs" for coding educational programs. Taking one of the utility components, a shuffle component, as an example, this component may be used when a teacher makes a quiz program. Students may memorize the location of the correct answer from a multiple-choice question if the same quizzes are repeatedly given to them in same order. With the shuffle component, the teacher can produce quizzes which randomize the correct answer within a larger set of wrong answers and show the correct answer out of a subset of multiple choices (say, one correct answer in four available choices), at random, with very little required coding.

High-level components

High-level components are made by assembling basic components. The target group for the high-level components is the beginner group, and each higher-level component is additionally able to function as a standalone application when it is compiled. The advantages of providing such programs as component sets, rather than as finished applications is as follows:

1) To be able to modify layout and properties as an individual teacher prefers, which can thus provide teachers with fulfillment of their specific pedagogical goals.
2) To be able to add new features by integrating additional basic components.
3) To be able to modify the behavior of the components with inherited functions of object programming.
4) To be able to produce an application program of their own design by putting together one or more high-level components.

We would like to restate two important points, mentioned above: 1) high-level components are, virtually, standalone applications, and 2) in a componentware approach the overall process should be focused on a specific educational field. Therefore, we focused on EFL (English as a Foreign Language) education, which often utilizes computers for educational use.

We have six types of high-level components now. Three of them can be used for creating Web-based quizzes (these three components can be used not only in EFL fields, but in other fields as well). Two of these components can be used for creating TOEFL(Fig. 4) and TOEIC practice programs, and one of the components can be used for creating a presentation program (which allows for slides, sound/movies and simple animations).•

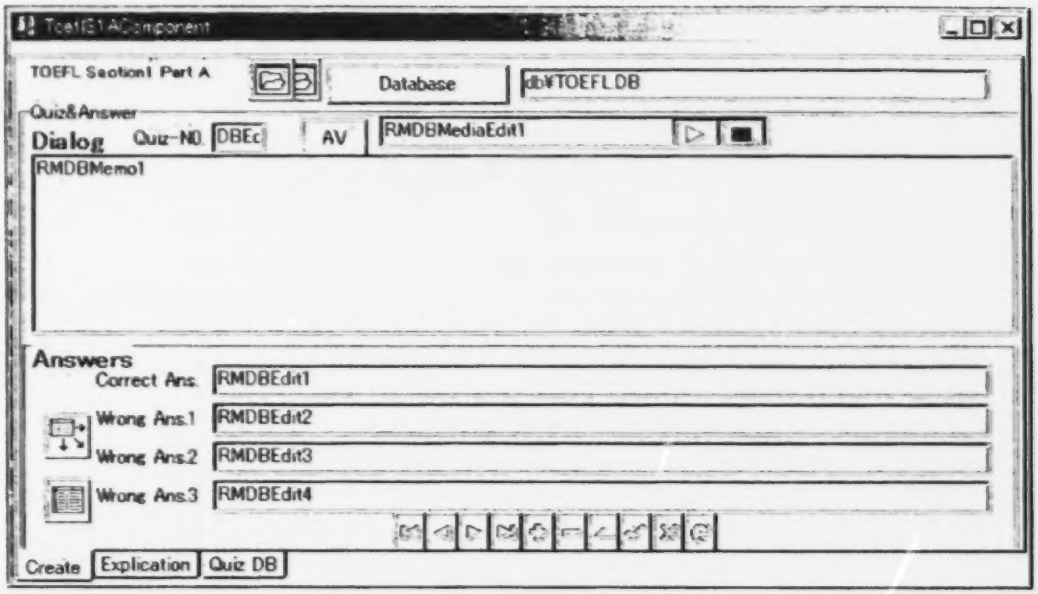

Fig 4. TOEFL Component (an Example of High-level Components)

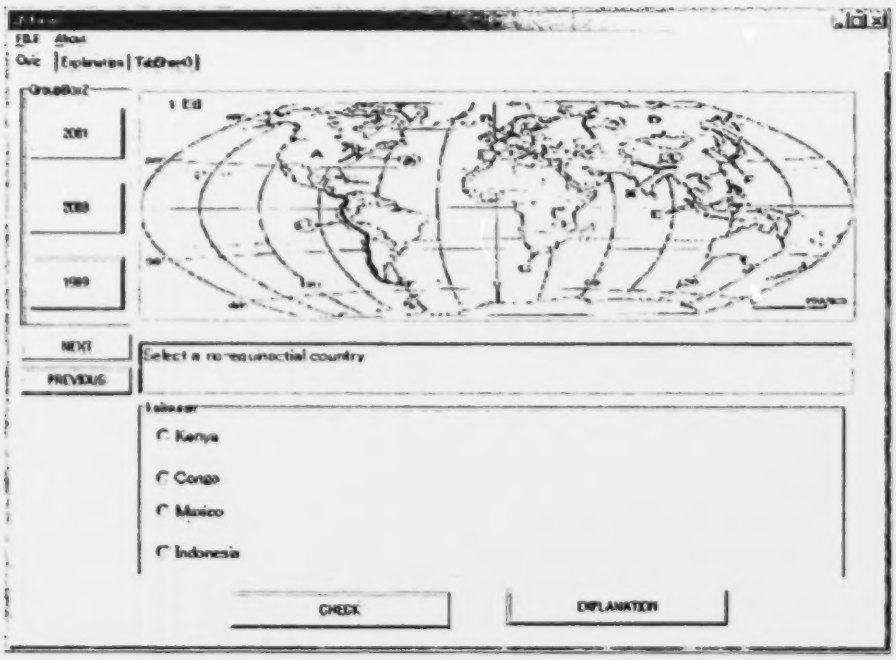

Fig 5. An Example of Application Programs made by students

BEST COPY AVAILABLE

All of these components are assembled with database components, but teachers do not have to have any knowledge of database programming. High-level components are able to manage all of these database-programming routines.

Example applications using our components

We have had our students make several example-programs using our components(Fig. 5). All of these programs have been made with very little coding. If the students had attempted to produce similar programs without utilizing the componentware design system, it would have required a great deal of effort: they would have needed to write a several-hundred lines of code, including condition- and repetition-program structures, and also would have to study and gain experience in API (Application Program Interface) concepts. Pragmatically, this is an unlikely prospect for any beginner programming-group. Using componentware, they were able to produce their own programs in approximately one week, with most of the students' design time spent on educational content-design.

Discussion

We have developed more than 30 components focused on EFL education, and have made several sample applications using those components (Fig. 3). We have been able to reduce the large amount of development time typically required for producing such applications, as well as reduce the level of knowledge required to create such applications. We do not think however that 30 components are sufficient for producing a large variety of EFL applications. We will need to continue to survey teachers' requests, and develop more components. Additionally, we have to further research each component's "hidden programming-routine size" (the number and "depth" of internal structures determining available features in a component). We can incorporate a large number of features into one component, however it is sometimes hard, particularly for the beginner group, to become skilled in the use of a more complex component, as there tends to be a larger number of parameters to set in a feature-laden component. This fact has shown us that an "all-purpose" tool (with features for every contingency) may sometimes be functionally useless for its target users. On the other hand, if every individual component is too "simple", this creates a situation where a welter of components must be used in order to produce applications; as a result, it becomes difficult to decide which component to select and as well, more coding becomes necessary. Thus, there is a trade-off between the 'simple component' approach versus the 'multi-purpose component' approach, which must be balanced.

Conclusion

We have described a new approach to educational software development. Teachers should be able to accomplish the following goals utilizing a componentware approach:

For both Beginner and Intermediate Groups:

1) To make new applications for the classroom using high-level components.
2) To make new and sophisticated applications for their classrooms mixing high-level-components and basic-components

For Intermediate Groups:

3) To make new types of high-level components and/or applications using basic components, and to provide them to the beginner group.
4) To make new basic components using/improving our components, and to provide them to the beginner group.

As previously mentioned, we have been focusing on EFL education, however we believe our development concept is applicable to numerous educational fields. We have recently begun to develop basic components for mathematics and physics education. The Borland Delphi object-Pascal language is our chosen development environment. Nevertheless, the componentware approach can be executed in other object-oriented language environments.

References

Aoyama, M. (1966). Componentware: Building Applications with Software Components. *Information Processing Society of Japan*, 37(1), 71-79.

Jacobson, I., Christerson, M., Jonson, P., Overgaard G. (1995). Object-Oriented Software Engineering: A Use Case Driven Approach. Addison Wesley.

Jacobson, I., Booch, M., & Rumbaugh, J. (1999). *The Unified software development Process*. Addison Wesley.
McCarthy C., Matsuno, R., & Swan, K. (1997). Pronunciation Power: A Hypermedia Pronunciation Program For Learners Of English As A Second Language. *Proceedings of ED-MEDIA & ED-TELECOM 97*, 1312-1313.

Matsuno, R., Tsutsumi, Y., & Ushijima, K. (1999). A-MATE: A multimedia authoring system for teaching ESL, *Proceedings of ED-MEDIA & ED-TELECOM 99*, 237-238.

Gilbert, R. & Matsuno, R. (2000a). A Computer-Based Vocabulary Resource Database for JSL/EFL Education. *The Japan Association for Language Teaching 24* (9) 15-19.

Gilbert, R & Matsuno, R. (2000b). Complementing University EFL Science Courses: A CALL Multimedia Concordance Design for Vocabulary Comprehension. *Administration* 7 (1) 69-100.

Potter A., Kopp, G., Green, C. (1947): *Visible Speech*. Van Nostrand.

Udel, J. (1994). Componentware. *Byte*, 19(5), 46-56.

MEXT (Ministry of Education, Culture, Sports, Science, and Technology) (2001): Investigation into Actual Applied Computer-Use in K-12 Classroom Education [Japanese]. Available: http://www.mext.go.jp/b_menu/houdou/13/09/010911/main.pdf (Accessed March 4, 2002).

U.S. Department of Education
Office of Educational Research and Improvement (OERI)
National Library of Education (NLE)
Educational Resources Information Center (ERIC)

NOTICE

Reproduction Basis

X	This document is covered by a signed "Reproduction Release (Blanket)" form (on file within the ERIC system), encompassing all or classes of documents from its source organization and, therefore, does not require a "Specific Document" Release form.
	This document is Federally-funded, or carries its own permission to reproduce, or is otherwise in the public domain and, therefore, may be reproduced by ERIC without a signed Reproduction Release form (either "Specific Document" or "Blanket").